Управление потоками заданий в PostgresPro Enterprise

PgConf 2018

Фролков Иван, Постгрес Профессиональный

Потоки заданий

- Множество транзакций, которое необходимо выполнить определенным способом
- Задания
  - Могут выполняться длительное время с длительными паузами
  - Не могут быть потеряны
  - Не могут быть выполнены более одного раза при соответствующем завершении
  - Выполнены строго в определенной последовательности
  - Устойчивы к сбоям
- Только СУБД!

Одноразовые задания

- Создаются для разового выполнения
- Могут быть выполнены успешно или неуспешно
- Успешно выполняются гарантированно один раз
- Могут сами запросить свое повторное выполнение
- Могут быть строго упорядочены

Изменение позиций прайс-листа в отделениях

- Более восьмисот отделений
- 50-200 и более изменений позиций в день
- Требуется обработка данных перед отправкой — некоторые отделения не могут видеть изменения некоторых позиций
- Соблюдение строгого порядка (сначала добавляем позицию, потом ее изменяем, а не наоборот)
- Корректное подключение нового отделения
- При изменении позиции она должна быть изменена во всех соответствующих отделениях
- Устойчивость к сбоям
  - Если обновление не удалось выполнить, необходимо иметь список отделений, где это не удалось выполнить и по какой причине
- После окончания процесса необходимо проставить отметку «отправлено в отделения»

Варианты решения

- Логическая репликация
  - В 10 версии потребуется более 800 одинаковых таблиц, по числу отделений
  - Число соединений с сервером
- Сторонняя очередь
  - Сложности с упорядочением — нет гарантий строгого порядка
  - Требование двухфазной фиксации
  - да, poison message — в данном случае нормальное и правильное явление
- Одноразовые задания
  - Таких проблем нет. Вообще и совсем

Одноразовые задания

- 800*200=160000 заданий в день. Не так уж и много для БД
- Обработка делается тривиально (ну, если она сама тривиальная :-)
- Строгий порядок и определение полного окончания — интересный вопрос. Есть два варианта:
  - Каждое задание ставит на выполнение следующее. Требование — полный список заданий должен быть известен заранее или надо организовывать какую-то таблицу и т. п. В принципе решаемо…
  - Для задания можно определить, что оно должно быть выполнено только после некоторого другого

Пример - отправка транзакции в блокчейн (да и не только)

- Задача — необходимо отправить некоторую сумму на некоторый адрес
- **До блокчейна еще не добрались, но проблемы уже есть: как обеспечить однократную отправку?**

```
begin;
  perform do_payout(....);
  update txn set state='payed'
where id=1234;
commit;
```

- Commit — ok
- Что делать в случае rollback?
- Что делать в случае сбоя?
  - Сбой, кстати, не всегда по питанию. Неудачный триггер на txn при массовом обновлении тоже может многое.

```
begin
  update txn set state='paying'
    where id=1234 and state='ready';
commit;


perform do_payout(....);


begin
  update txn set state='payed'
    where id=1234;
commit
```

```
begin
  update txn set state='paying'
    where id=1234 and state='ready';
commit;
```

Сбой!

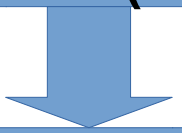
```
perform do_payout(....);
```

Сбой!

```
begin
  update txn set state='payed'
    where id=1234;
commit
```

```
begin
   update txn set state='paying'
     where id=1234 and state='ready';
commit;
```

```
perform do_payout(....);
```



```
begin
   update txn set state='payed'
     where id=1234;
commit
```

pgpro_scheduler

schedule.submit_job(query text [параметры…])

...

run_after timestamptz — выполнять после указанного момента времени

depends_on bigint[] - все перечисленные задачи должны быть завершены перед этой задачей

Отправка в блокчейн

- Bitcoind — REST API
  - Transaction sendtoaddress(Address address, Number amount)

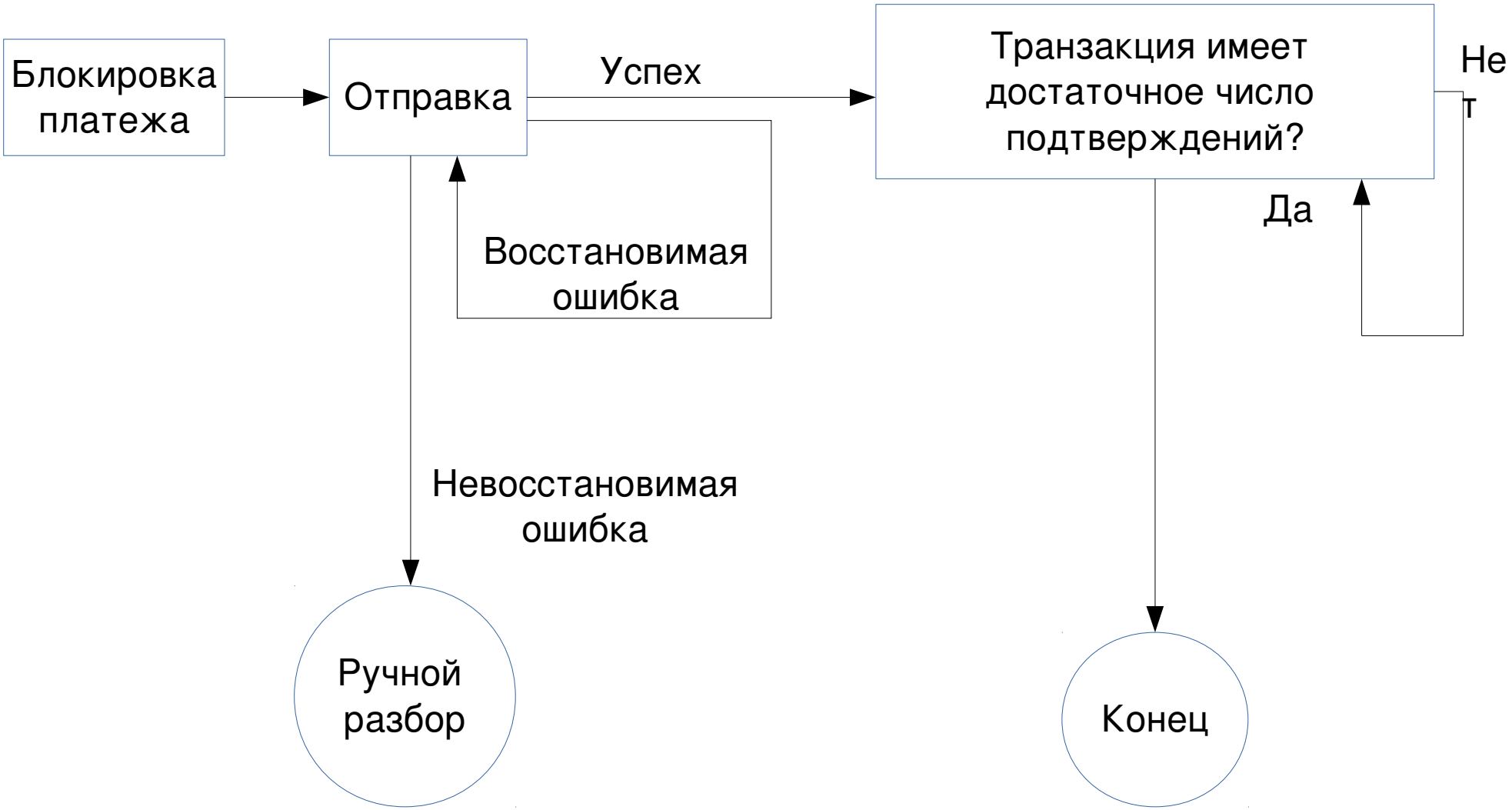
Отправка в блокчейн

- Успешно
  - На самом деле она попала в mempool
  - Надо дождаться, когда транзакция попадет в блокчейн и получит определенное число подтверждений
    - **Требуется периодически проверять число подтверждений транзакции**
  - Если не попала, отправить заново
    - **Требуется уметь повторять процесс заново**

Отправка в блокчейн

- Отправка
  - Неуспешно
    - **Нет средств — через какое-то время попытаться отправить заново**
    - **Сетевая ошибка — также попытаться переотправить**
    - **Таймаут — инициировать процесс ручного разбора**
- Надо уметь классифицировать ошибки на неисправимые и те, которые могут не повториться

Процесс выполнения отправки



pgpro_scheduler

- schedule.submit_job(query text [параметры...]) returns bigint
  - params text[] - доступно как $1, $2..
  - **run_after** timestamptz – когда запускать
  - **depends_on bigint[] – от каких задач зависит**
  - **…**
- schedule.get_self_id()
- schedule.cancel_job(job_id bigint)
- schedule.resubmit(run_after interval default NULL)
- schedule.deactivate_job(job_id integer)
- schedule.activate_job(job_id integer)
- schedule.drop_job(job_id integer)
-

Почему pgpro_scheduler

- Во-первых, потому что он уже есть
- Во-вторых, какие есть альтернативы?
  - Oracle DBMS_SCHEDULER
    - требуется собственно Oracle
    - Несколько дубово, на мой, возможно ошибочный, взгляд
  - SQL Server jobs
  - jBPM, Activiti, Camunda и др. средства BPMN
    - это Java/C#, GUI, специфическая среда выполнения, Spring/JEE/C#
  - Apache Camel
    - Тоже Java и Spring. Крутая learning curve, сильно на любителя
  - Spring Batch/JEE Batch
    - Крутая learning curve, очень сильно на любителя
  - Celery
    - Python, RabbitMQ, нетранзакционно, опять-таки на любителя
  - Gearman
    - Нетранзакционно
  - При желании можно написать свое. Все, в принципе, можно написать...

Вопросы?